資訊管理學系 陳士杰老師

作業系統
Operating Systems

概說
Overviews

國立聯合大學
NATIONAL UNITED UNIVERSITY

■本章重點架構

- O.S.的目的
- Bare Machine vs. Extended Machine
- Buffering (緩衝)
- Spooling (線上同時週邊處理)
- I/O Bound Job 與 CPU Bound Job
- System Types
  - Multiprogramming System (多元程式規劃系統)
  - Time Sharing System (分時系統)
  - Distributed System (分散式系統)
  - Multiprocessor System (多處理器系統)
  - Real Time System (即時系統)
  - Clustering System (集成式系統)
  - Handheld System (手持系統)
- Computation Mode (Environment)

■O.S.的目的 (什麼是作業系統)

- **使用者觀點:**
  - 做為電腦使用者 (User) 與電腦硬體 (Hardware) 之間的介面，使得User易於使用Hardware
  - 提供一個讓User Program易於執行的環境
- **系統觀點:**
  - 是一個資源分配者 (Resource Allocator)
    - 解決資源使用上之衝突
    - 讓資源公平且有效的被利用
      - HW: CPU, Memory, I/O Device
      - SW: Share Files
  - 監控User Program的執行，以防止不正常的運作造成對系統的危害

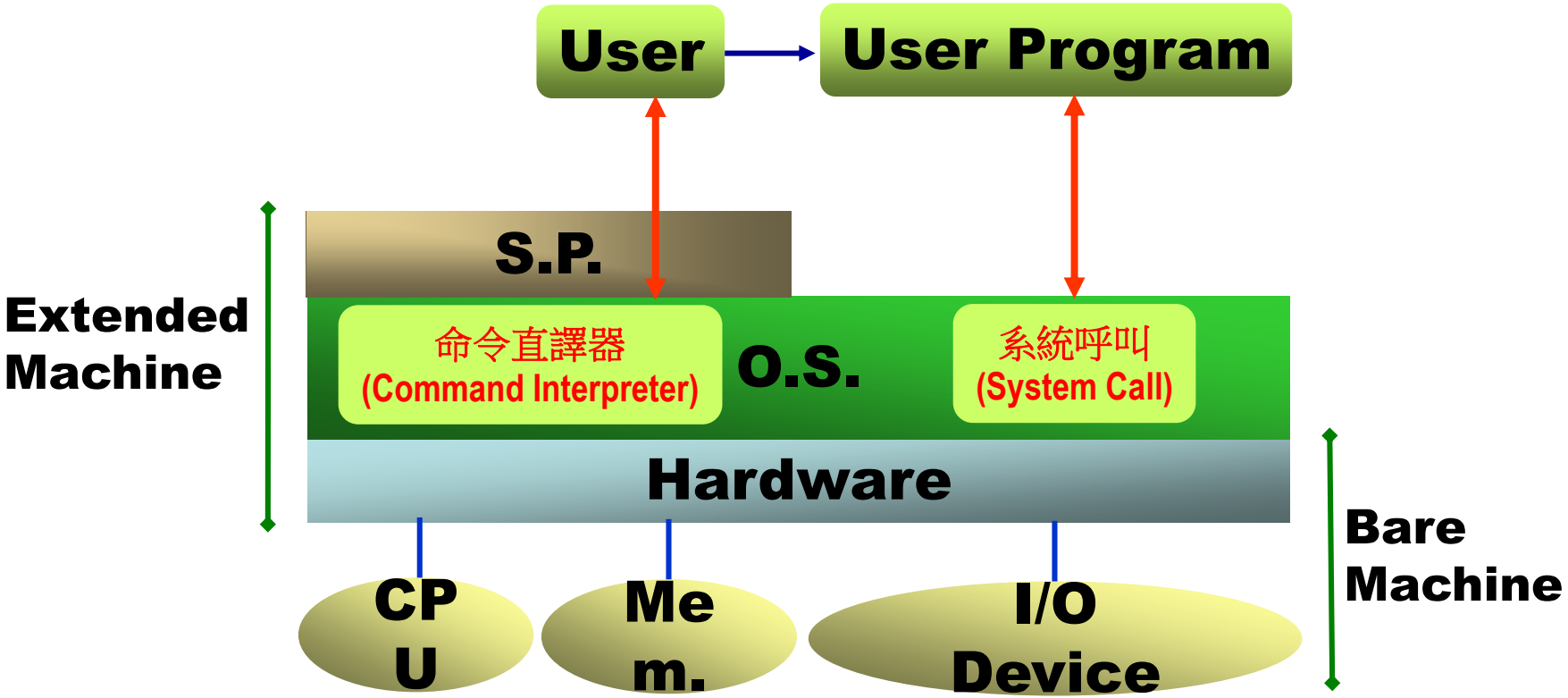
OS服務的對象有二：
- User (人)
- Program

■O.S.的目標

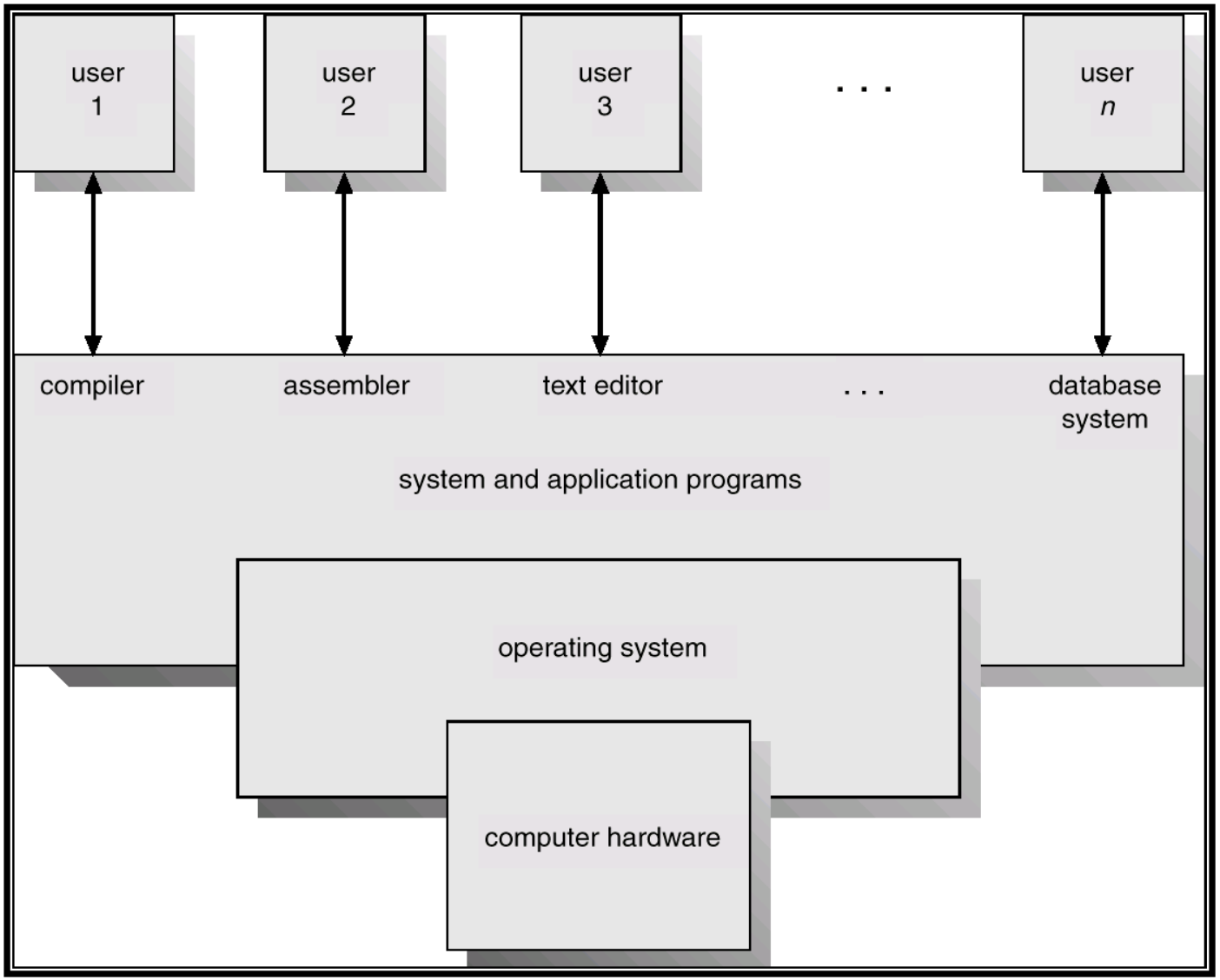
- 讓電腦系統在使用上很方便–**創造使用者的便利**。
- 以效率較高的方式來使用電腦硬體–**提高電腦系統的效率**。

Bare Machine vs. Extended Machine

- **Bare Machine (裸機)**
  - **Def:** 由**硬體元件**所組成，沒有任何附加的系統軟體。
  - 對使用者使用非常不方便!! (˙.˙要自已寫機器碼)
- **Extended Machine (延伸機器)**
  - **Def:** 在**Bare Machine**上，加上系統軟體 **(e.g., O.S. etc.)**，構成**Extended Machine**。
  - 從硬體上一直加上**方便使用者使用**的軟體。



- O.S.服務的對象有二:
  - User
  - User Program


user 1
user 2
user 3
. . .
user n
compiler
assembler
text editor
. . .
database system
system and application programs
operating system
computer hardware

■作業系統的演進

- 早期的系統
  - 是由**人工操作安排工作順序**。
  - 只有**電腦硬體**，程式人員寫出程式後，將程式交給操作員經由**控制台**來執行。

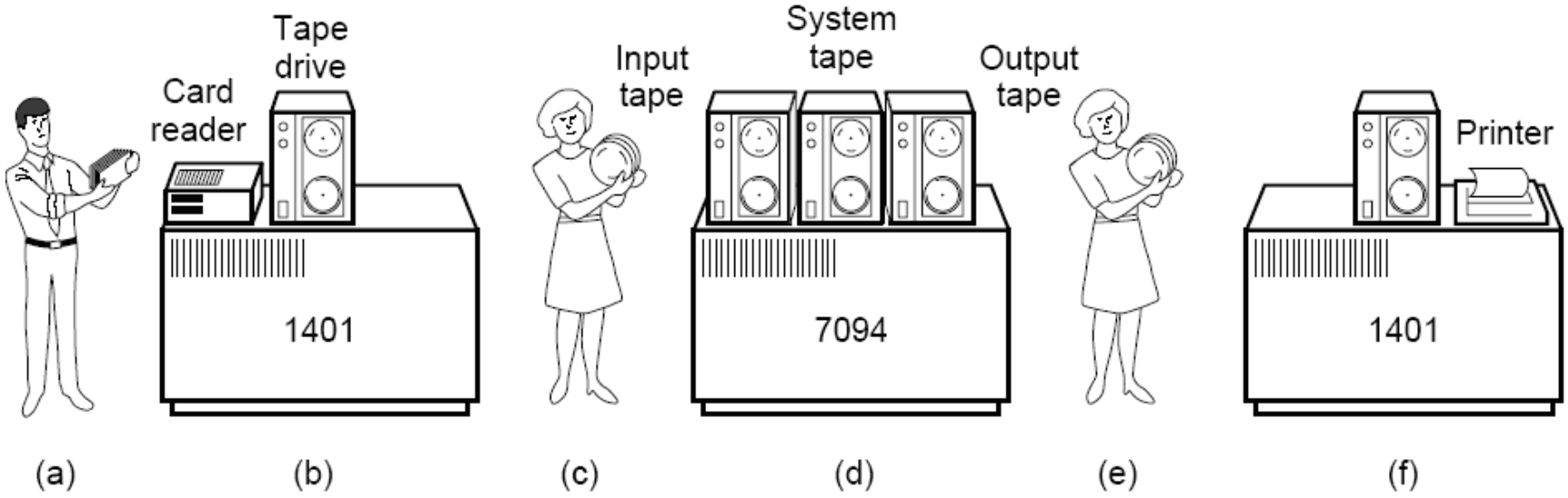
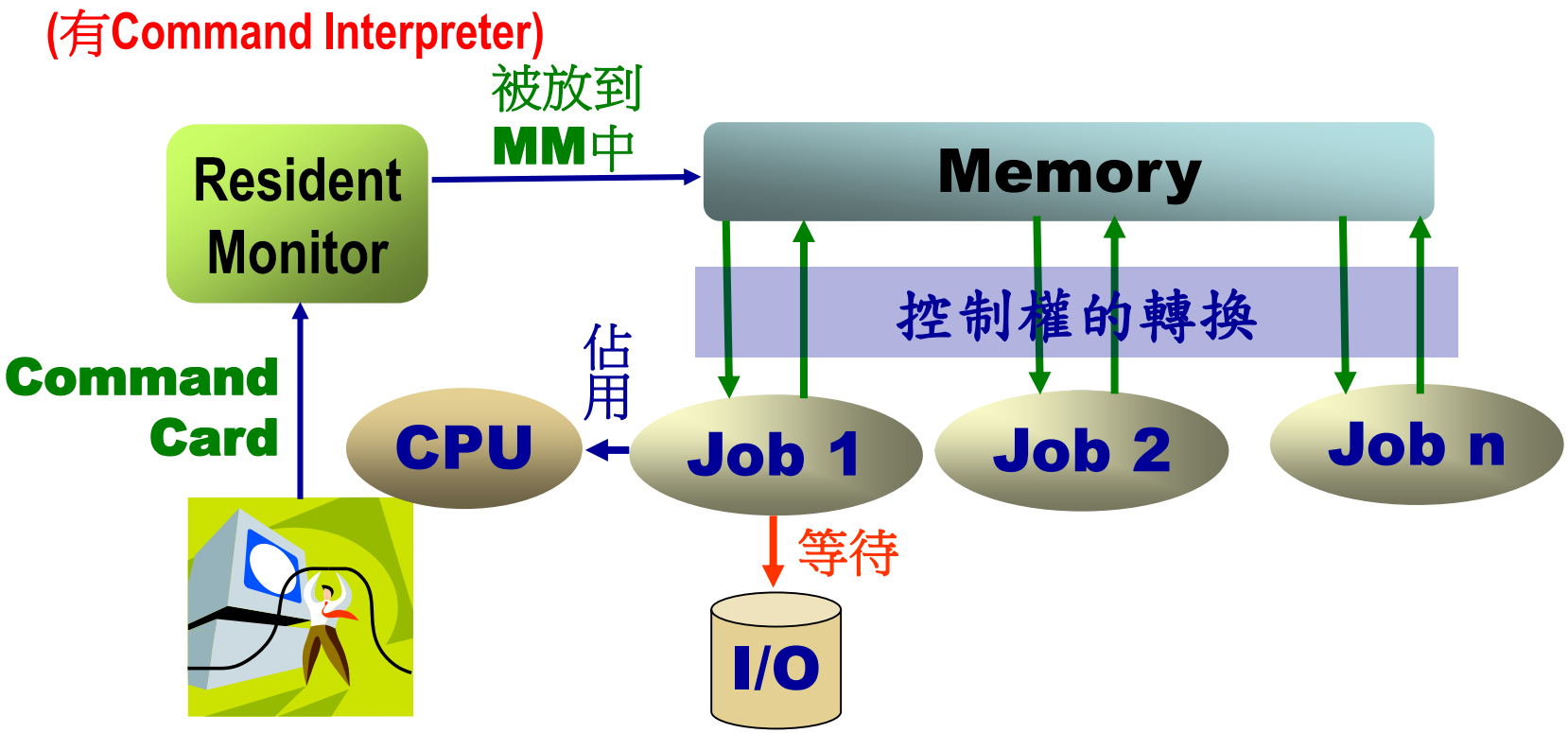


Figure 1-1. An early batch system. (a) Programmers bring cards to 1401. (b) 1401 reads batch of jobs onto tape. (c) Operator carries input tape to 7094. (d) 7094 does computing. (e) Operator carries output tape to 1401. (f) 1401 prints output.

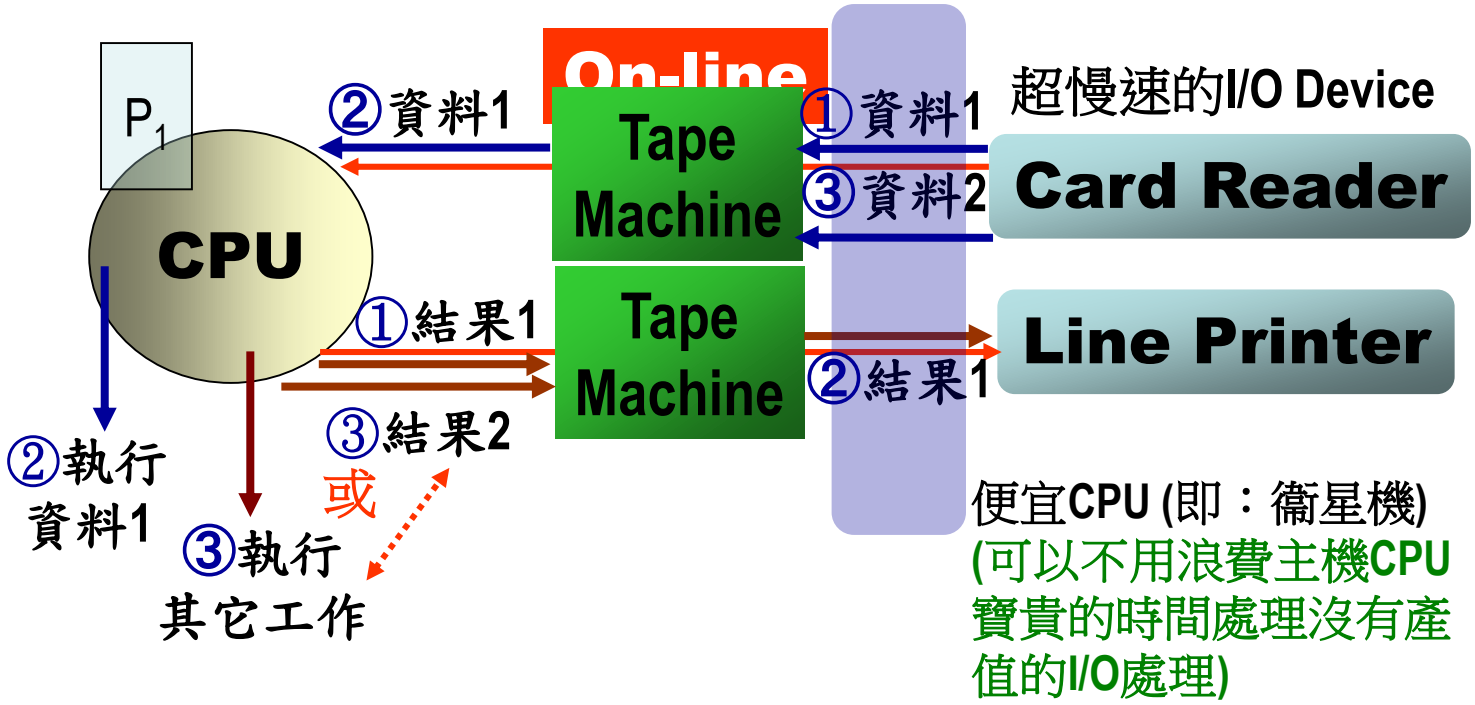
早期系統所引發的問題

- **CPU Idle Time 過長**，造成User完成Job的時間很難預估。
- 原因：
  - 人為介入的時間過長 (即：人為Set-Up時間太長)
  - I/O Device運作速度太慢，導致CPU被迫等待I/O運作完成，使得CPU Idle Time過長 (即：I/O速度<<CPU速度。∵ I/O Device有機械特性)
- 解決方法：
  - 問題一：利用**Resident Monitor (常駐監督程式)**
  - 問題二：
    - 以**較快速的設備**介入CPU與慢速I/O Device之間(CPU與I/O同時運作)
      - **Off-line**
      - **Buffering**
      - **Spooling**
    - 讓CPU**保持Busy**
      - **Multiprogramming**

Resident Monitor

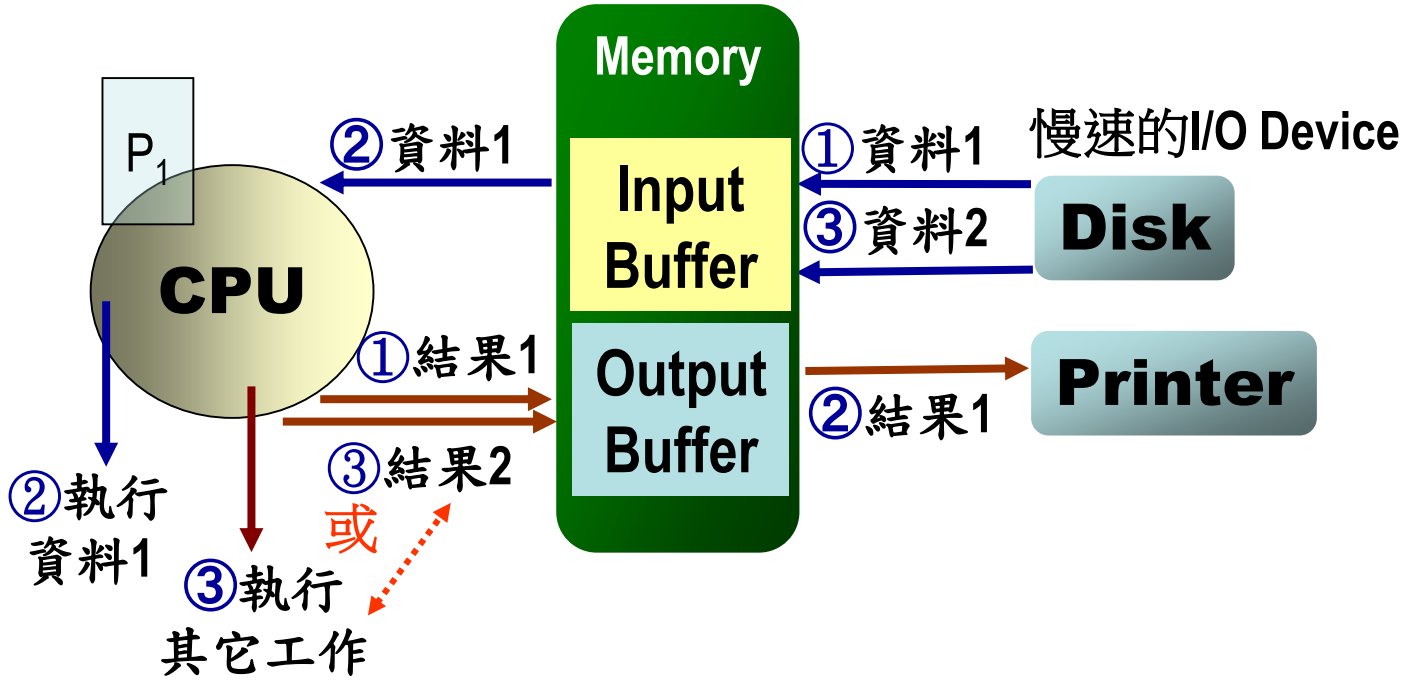


Off-line



- ② 與 ③ Overlay Execution。
- 如何讓Tape Machine與超慢速I/O Device溝通?
  - 特用的I/O程式或軟體
  - 便宜的CPU

Buffering



- ② 與 ③ Overlay Execution。
- 若I/O運作可以及時將下一波處理的Data往Input Buffer送，則CPU可避免Idle (理想)。

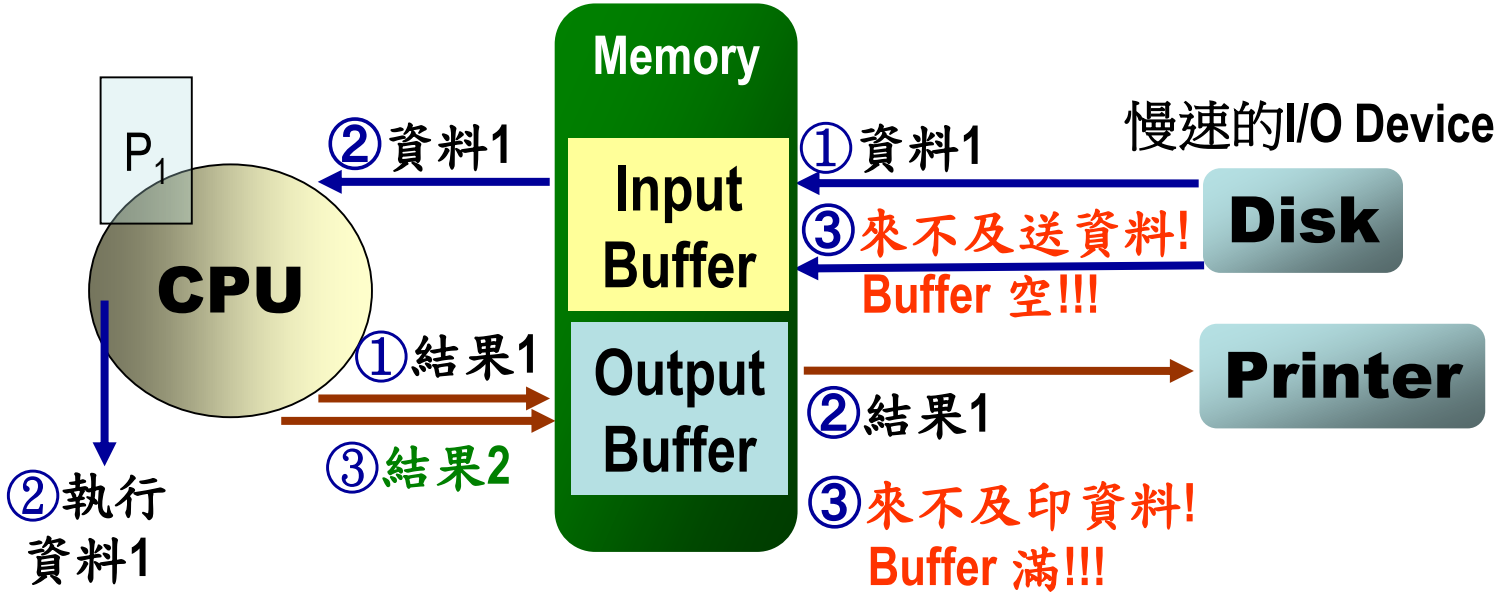
系統內運行的工作有兩類

- **I/O Bound Job**
  - **Def:** 此類型的工作包含**大量的I/O Operations**，所以需要**大量的I/O Operation Time**，對於CPU Computation Time的需求量較少。
  - 工作的效能是取決於**I/O Device的速度**。
- **CPU Bound Job**
  - **Def:** 此類型的工作包含**大量的CPU Computation**，所以需要**大量的CPU Computation Time**，對於I/O Operation Time的需求量較少。
  - 工作的效能是取決於**CPU的速度**。

I/O Bound Job 於 Buffering

- 若I/O Bound Job放到Buffering中執行，則：
  - CPU總是面對空的Input Buffer而被迫等待。
  - CPU總是面對滿的Output Buffer而被迫等待。

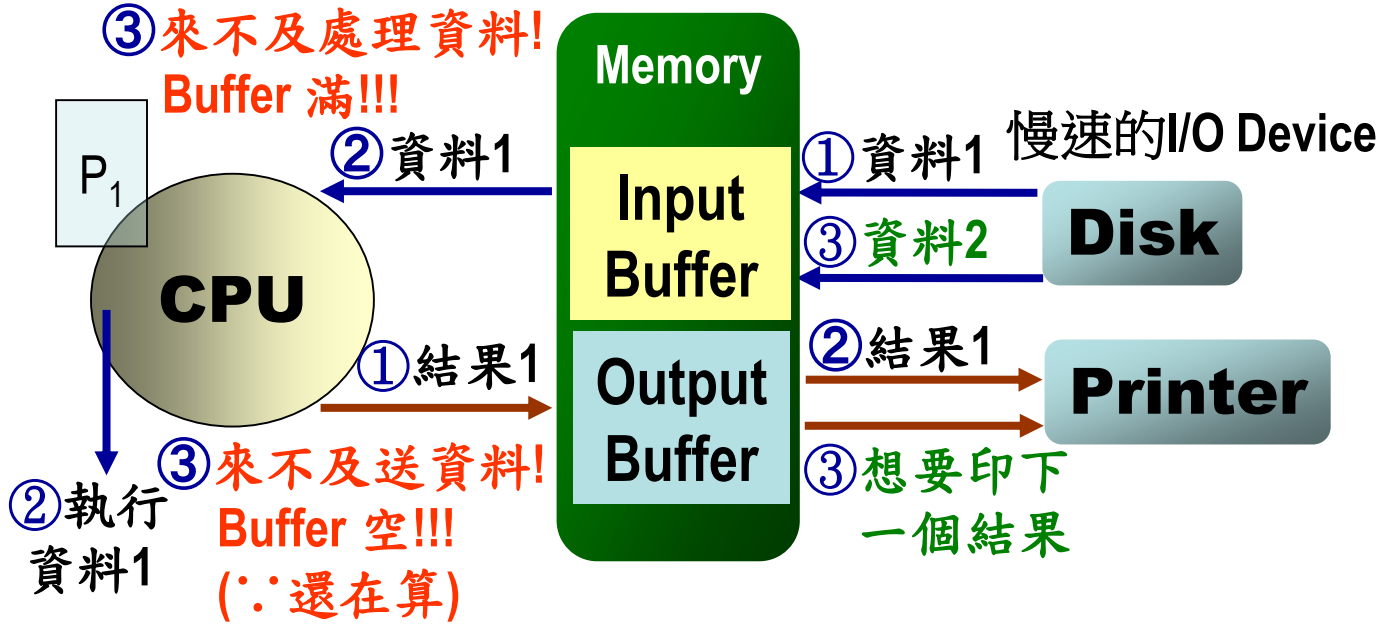
∴CPU常常Idle。



CPU Bound Job 於 Buffering

- 若CPU Bound Job放到Buffering中執行，則：
  - I/O Device總是面對滿的Input Buffer而被迫等待。
  - I/O Device總是面對空的Output Buffer而被迫等待。

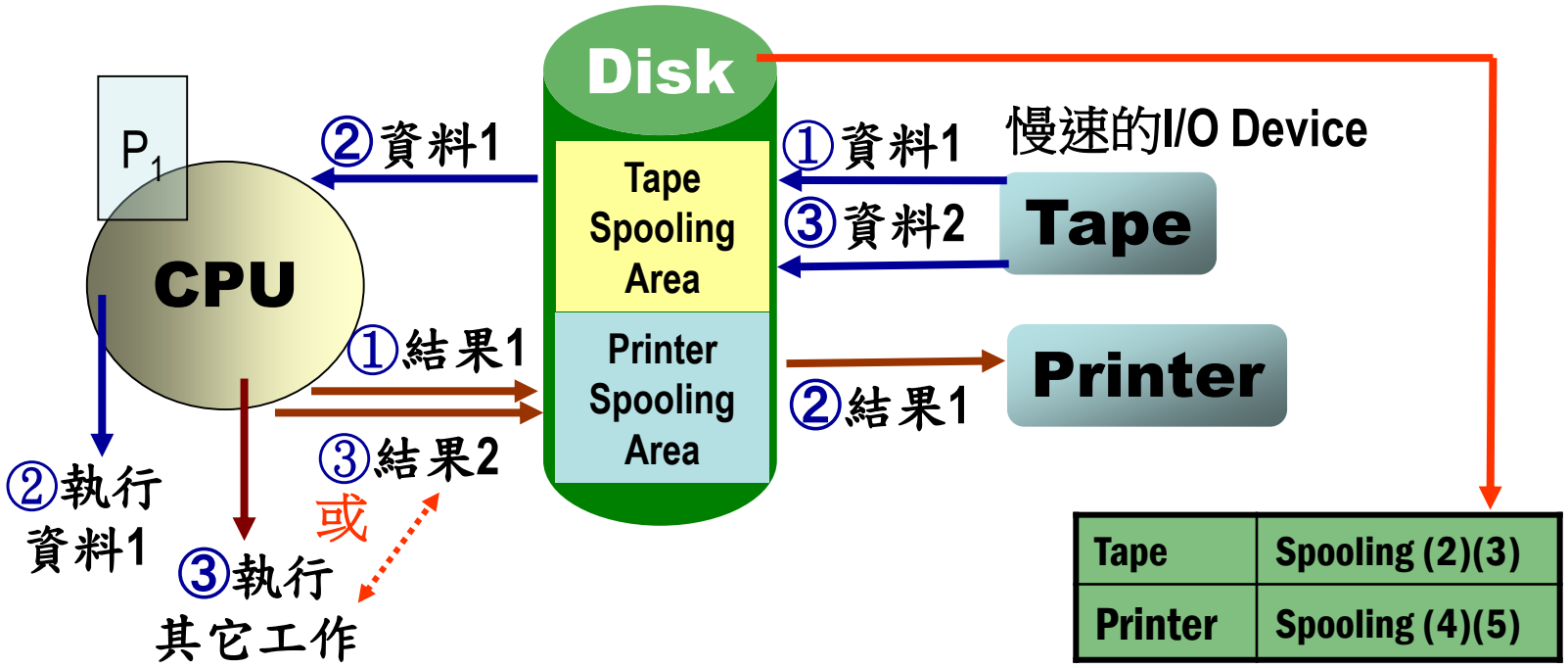
∴I/O Device常常Idle。



- 因此，由資源利用度來看：

系統內I/O Bound Job與CPU Bound Job
必需能均勻混合

Spooling (Simultaneous Peripheral Operation On- Line Processing) 線上同時週邊處理



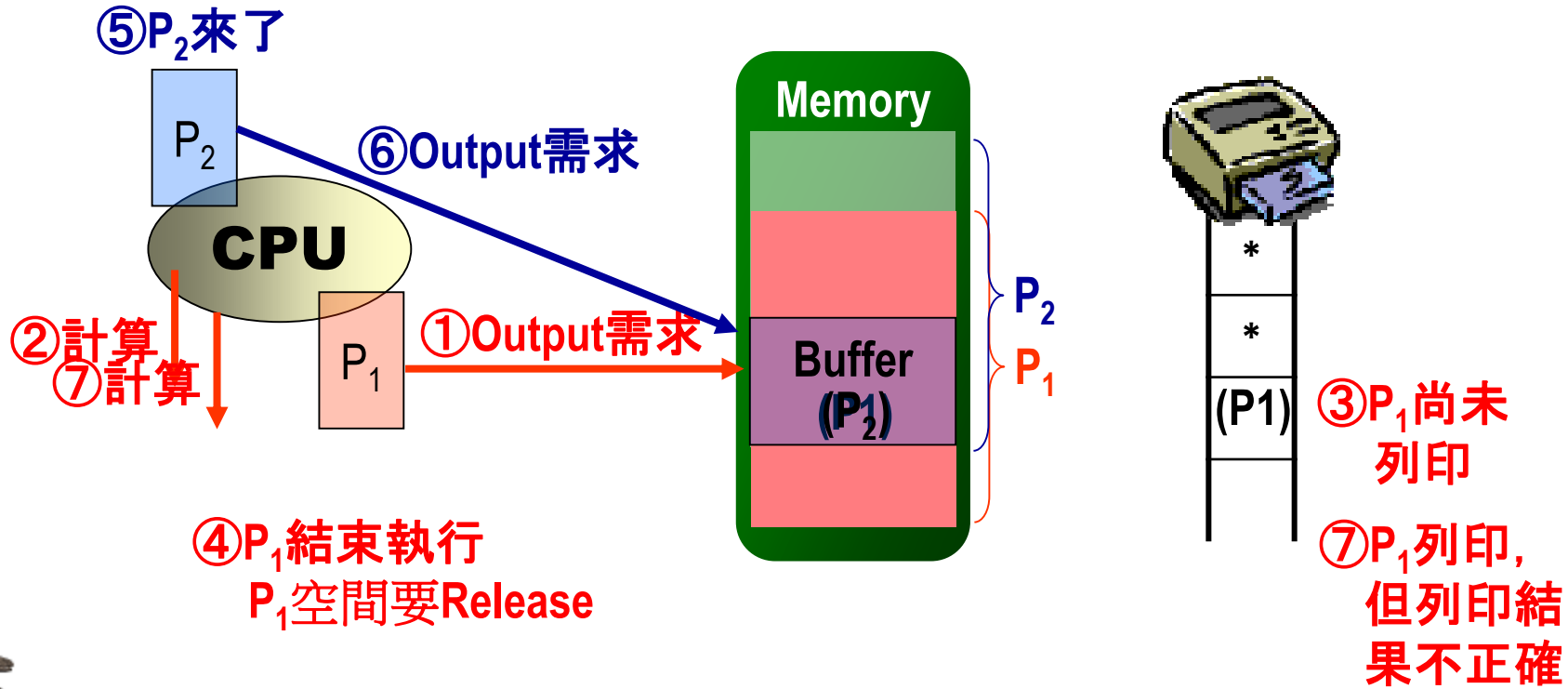
- ② 與 ③ Overlay Execution。
- O.S.會在Disk中花費少許空間建立一個Table來記載Input /Output Data是在Disk中的哪一個Spooling的哪一個區域。

- **Def:** Spooling是把磁碟 (Disk) 當成一個**巨大的緩衝區**在使用。在Disk中有一個區域為**Spooling Area**，每個I/O Device可以有自已的Spooling Area。
  - 對於**輸入**，Input Device (e.g., Tape)可將Input Data先行送入Disk的Spooling Area。當CPU從Spooling Area取出Input Data在計算時，Input Device也可以同時將下一波處理的Input Data往Spooling Area傳送。
    - ∴ **CPU Computation與I/O Operation可以重疊執行**。
  - 對於**輸出**，CPU將Output Data製成Print File，輸出到Spooling Area中，CPU即可往下繼續其Computation的工作，而此時Printer亦可同時列印。

- Spooling的優點：
  - 可讓不同工作的CPU Computation與I/O Operation同時執行，使CPU與I/O Device同時保持忙碌。
  - 提升Resource Utilization與Throughput。
- 缺點：
  - O.S.需要少量的Memory空間用來記錄各I/O Request之執行狀況，以及一些Disk空間來存放所需之Table。

Spooling與Buffering的不同

- Spooling允許不同Job之間的CPU Computation與I/O Operation可以重疊執行。
- Buffering僅能讓同一Job的CPU Computation與I/O Operation重疊執行。
  (前題:假設一個Program 的I/O Buffering區域是在Program 自已的 Memory Space)



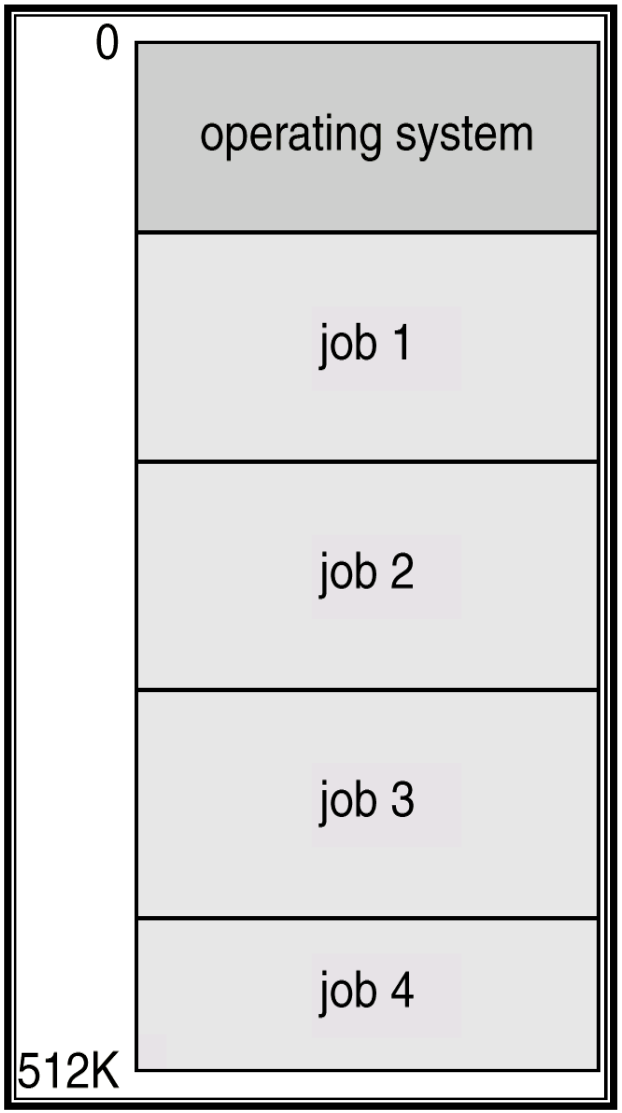
- P1的Output工作與P2的計算工作重疊執行，但結果不正確。
- 解決方法：
  - **Lock User Memory Space**
    - 把**整個Process** 之Memory區域鎖住，不準Swap
  - **Lock局部的Memory區域**
  - **把資料送往O.S. 所在的Memory區域**

系統類型 (System Type)

- Multiprogramming System
- Time-Sharing System
- Real Time System
  - Hard
  - Soft
- Distributed System
  - Loosely-Coupled (Distributed System)
  - Tightly-Coupled (Multiprocessor或Multiprocessing)
    - Symmetric (對稱式)
    - Asymmetric (非對稱式)
- Clustering System
- Handheld System

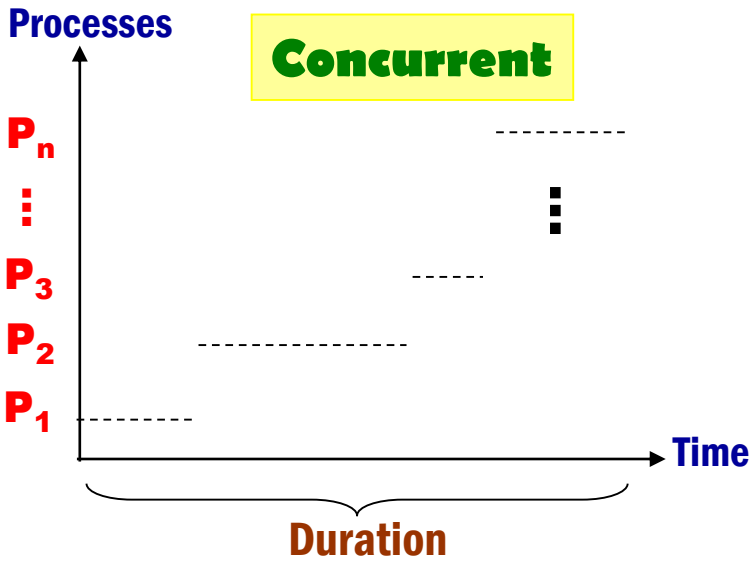
■多元程式規劃 (Multiprogramming System)

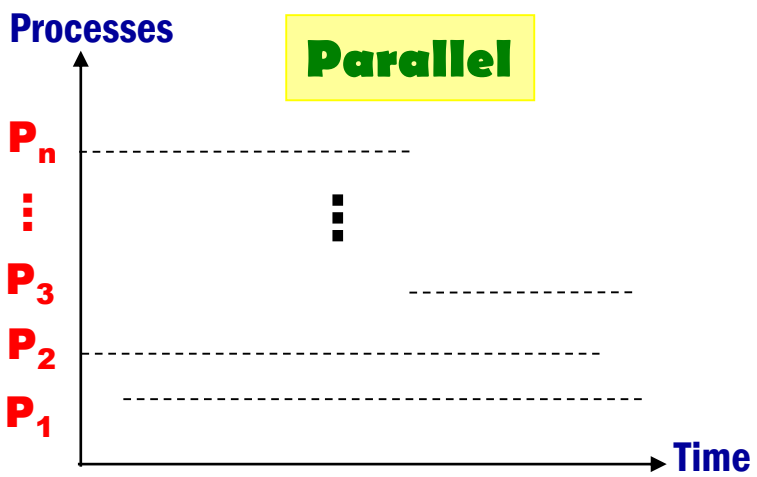
- **Def:** 系統 (或Memory) 中存在多組Process "同時執行"。
- 目的: 避免CPU Idle, 進而提升CPU Utilization。
- 作法:
  - 透過CPU Scheduling的技巧, 讓CPU在許多不同的Process之間切換(即: Context Switch), 使得CPU總是有事可做 (always busy)。
  - 當某個Process取得CPU正在執行時, 若因為某個事件發生 (e.g., wait for I/O complete...etc.) 而需被迫暫停時, 此時O.S. 透過CPU排班 (CPU Scheduling) 將CPU切換給其它需要的Process使用。因此, 若系統內存在許多欲獲得CPU執行的Processes, 則可以讓CPU always busy, 故CPU不致於Idle。



Process同時執行的方式

- **Concurrent (並行)**
  - 在**單一時間點**只有一個**Process**在執行!! 所強調的是**一段執行時間**內，有多**Process**同時執行，而非單一時間點。
  - 單一顆**CPU**即可做到。
  - **Multiprogramming System**即屬此類。
- **Parallel (平行)**
  - 在**單一時間點**有很多的**Process**在執行!!
  - 需多顆**CPU**方可做到。
  - **Multiprocessor System, Distributed System**





在Multiprogramming下之O.S. 需求

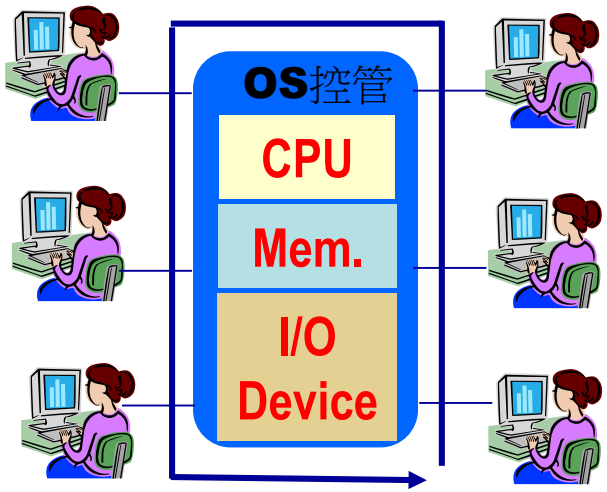
- 這類型的系統同時會有好幾個工作在記憶體中準備執行，此時，就要由作業系統選擇其中之一交由CPU來加以執行，這就是**CPU排程 (CPU Scheduling)** 的工作 。
- 這類型的系統可以將數個要處理的工作存放在記憶體中，以等待執行。但是，如果電腦系統所提供的記憶體容量不足以將使用者所要處理的所有工作全部存放起來的話，此時作業系統就要在這些工作中加以取捨，看是哪一些工作要先放入記憶體中執行，而另一些則在記憶體以外的輔助儲存體中暫時存放以等待處理。這種工作就是**工作排程 (Job Scheduling)** 。
- 在多元程式系統中，我們會將多個要執行的工作存放到記憶體中以便執行，所以需要**記憶體管理 (Memory Management)** 機制來管理這些存放在記憶體輪流執行的工作 。

Multiprogramming Degree

- 指:系統內所存在的待執行之Process數目。
- 一般而言，Multiprogramming Degree愈高，則CPU Utilization愈高。
  - 例外狀況: Thrashing (輾轉現象)
    - 當Multiprogramming Degree高到超過某個程度時，由於系統內Process數目愈多，會造成所有Process皆忙於從事置入、置出(Swap in, Swap out)於主記憶體及輔助記憶體(Disk)的動作，CPU反而因此而idle，造成CPU的利用度下降。

■分時系統 (Time Sharing System)

- **Def:**
  - 是Multiprogramming System的一種。
  - OS透過下列資源共享 (Resource Sharing) 的技術，使得每個User都認為有一套專屬的系統存在。
    - CPU Scheduling採用Round-Robin (RR) Scheduling，使User共用CPU
      - O.S.會規定一個CPU Time Quantum (Slice)，若Process取得CPU後，未能在此Quantum內完成工作，則必須放棄CPU，等待下一輪迴。
    - 所有User共享Memory Space
    - 透過Spooling共享I/O Device



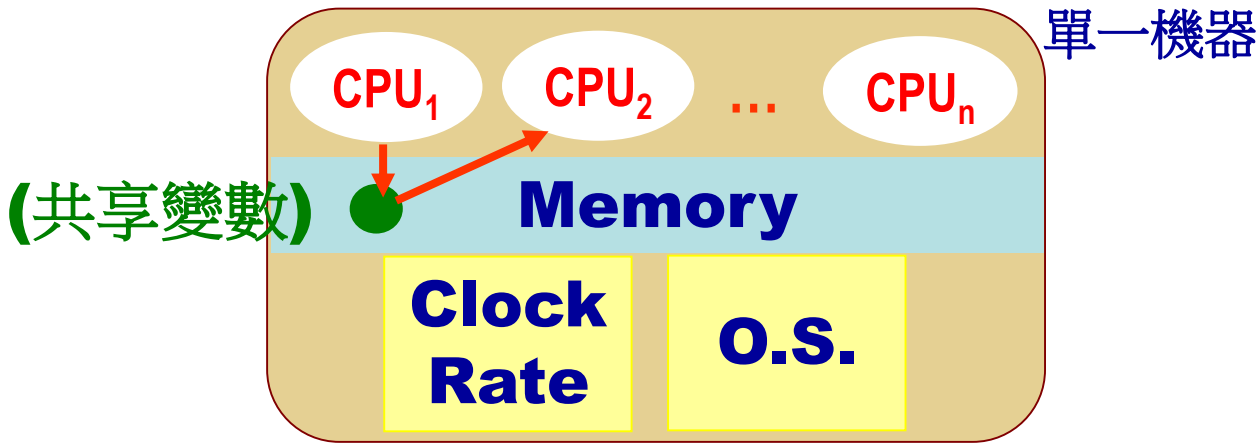
- 強調對所有User一律公平。
- 因為CPU一直不斷的在不同工作間來回切換，所以使用者可以和正在執行中的程式做交談，所以分時系統適用於交談式(Interactive或Hands-on)環境。
- 因為要求能夠即時對使用者的要求做回覆的動作，所以它的反應時間(Response Time)要很短，通常讓使用者等待系統回應的時間是設定在一秒以內。

■分散式系統 (Distributed System)

- 可分成兩種類型 :
  - **Tightly Coupled (緊密耦合)** : **Multiprocessor System, Parallel System.**
  - **Loosely Coupled (鬆散耦合)** : **Distributed System**

緊密耦合 (Tightly Coupled)

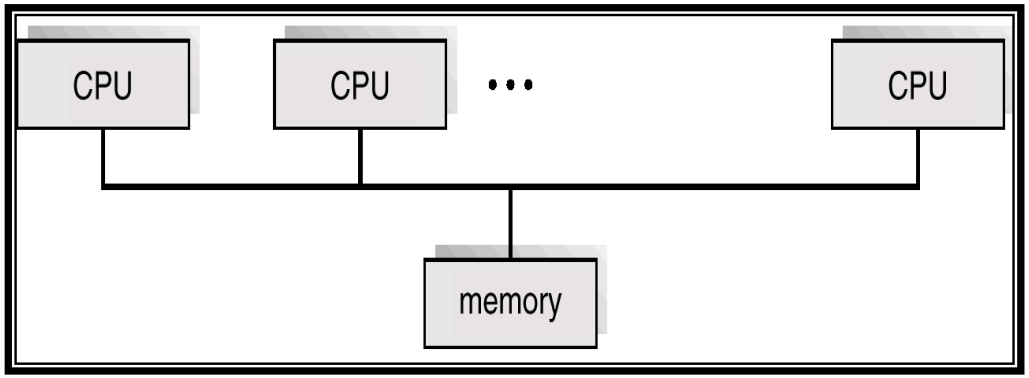
- **Def:** 同一部機器具有**多顆 (≥2) CPU** (或Processor)存在，且具有下列特徵:
  - 共享相同的**記憶體空間**、**I/O Device**、**Bus**
  - 受**同一個Clock及O.S.**的控制與協調
  - CPUs之間的溝通(即:Data的交換)大部份是採**Share Memory**技術。



- 也可稱為 Multiprocessor System (多處理器系統)、Multiprocessing System 或 Parallel System (平行系統)。
- Multiprocessor System可將許多Processes或一個Process上的許多Subtasks配置到不同的CPU上，以Parallel方式同時執行。
  - ∴支援Parallel Computing
- Multiprocessor System可再分為兩個類型：
  - Symmetric Multiprocessing (SMP): 對稱式多元處理
  - Asymmetric Multiprocessing (ASMP): 非對稱式多元處理

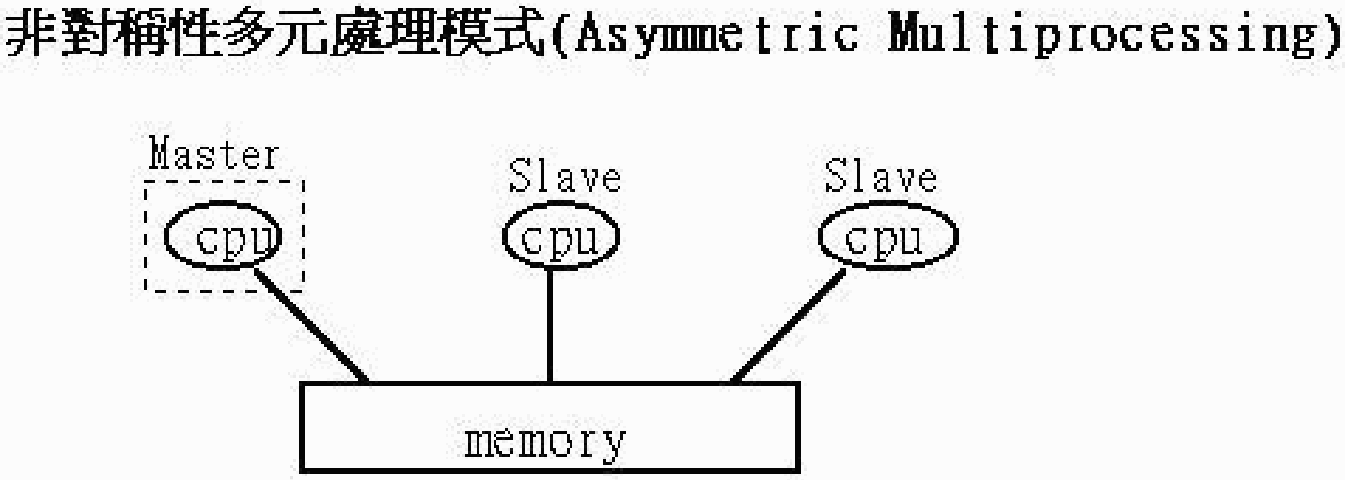
對稱式多元處理 (Symmetric Multiprocessing; SMP)

- 每一個Processor具有相同的功能。
- 可靠度 (Reliability) 較高，當某一個Processor壞了，則在其上未完成的工作可以轉移到其它的Processor繼續執行。∴系統不會整個Crash。
- 強調Load Balance (負載平衡)，希望在每一個Processor上的負擔均等。



非對稱式多元處理 (Asymmetric Multiprocessing; ASMP)

- 每個 (或某群) Processor 各自**負責不同的工作 (功能)**，通常具有一個**Master Processor**負責控制、協調及分配Process到其它的Processors去運作。其餘的Processors稱為**Slave Processor**。
- 又稱為**Master/Slave 架構**。
- 通常效能比SMP佳，但可靠度較差。



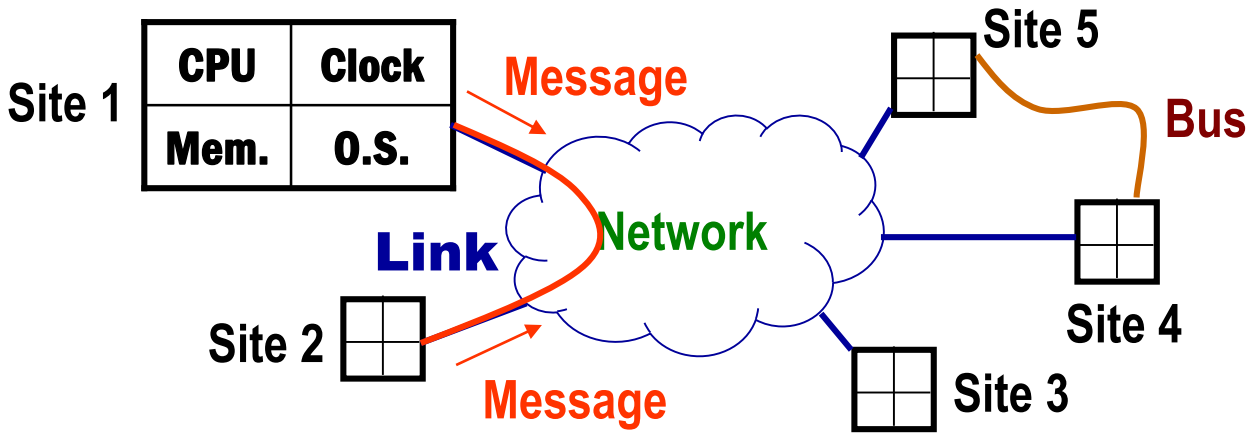
- **Multiprocessor System 的優點:**
  - **提升產能 (Increased Throughput)**
    - ∵可**同時執行多個工作**，或一個工作**拆成許多部份**，每個部份在**不同的CPU**上執行。
  - **合乎經濟效益 (Economy of Scale)**
    - ∵Processors共享**相同的Memory、I/O Devices、Bus**...等，故Cost低。
  - **可靠度增加 (Increased Reliability)**

有n個CPU，效能是否就為原來的n倍?

- 若有n個CPU，效能並非原來的n倍。
  - ∵CPU 之間的 **Communication (通訊)** 及 **Resource Contention (資源競爭)** 會產生額外的消耗。

鬆散耦合 (Loosely Coupled)

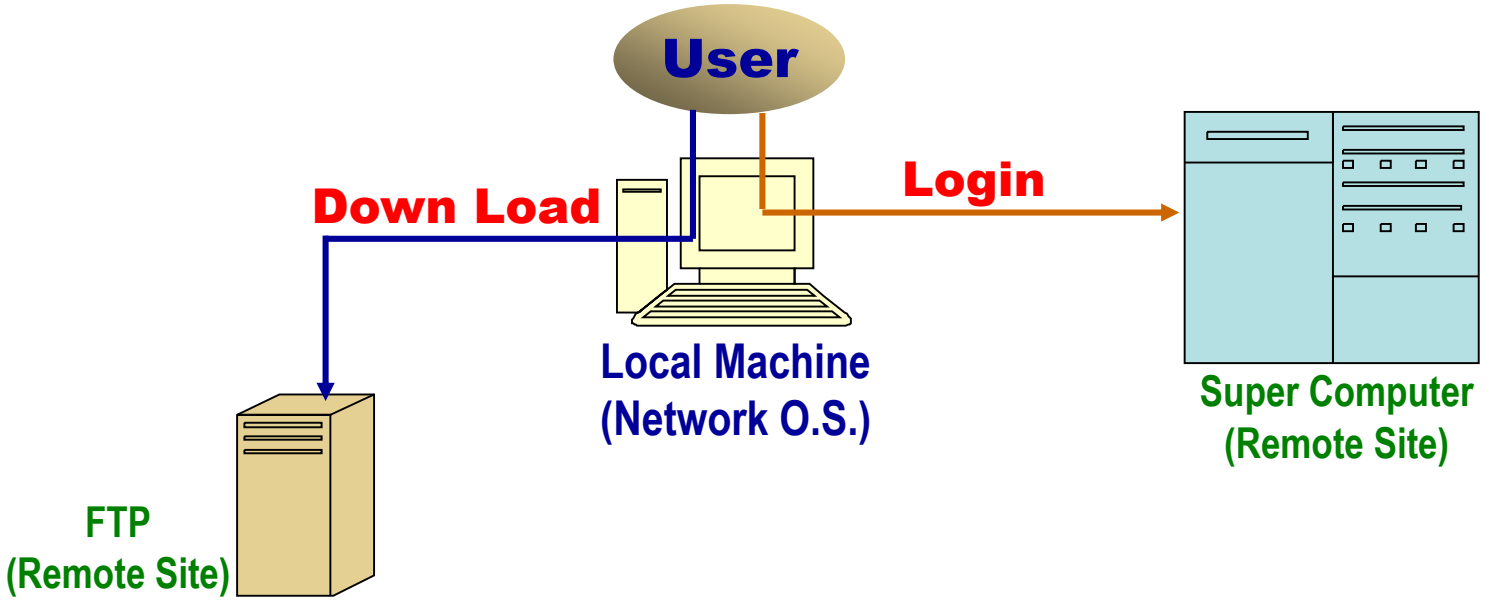
- 目前大多稱為**Distributed System**。
- 每一部機器都有**自已的Processor (CPU), Memory Space, I/O Device, Clock...**等。
- 通常不一定受同一個**O.S.**控制。
- 各個**Process**之間溝通是以 "**Message Passing**"為主。
  - 各個**Processor** (或機器)之間，彼此以**Network**或**高速Bus**進行連接，以達到**Message**交換的目的。



- 在Distributed System (Loosely Coupled)中，可分為兩種Modes:
  - **Client-Server Model**
    - **Client:** 發出Service Request的一方
    - **Server:** 提供Service的一方
    - e.g.,
      - Computing Server Model
      - File Server (e.g., FTP)
      - Print Server
      - Mail Server
      - Proxy Server
  - **Peer to Peer Model**
    - 無主控型式的Model
    - 每個Site可以是Client/Server

網路作業系統 (Network Operating System)

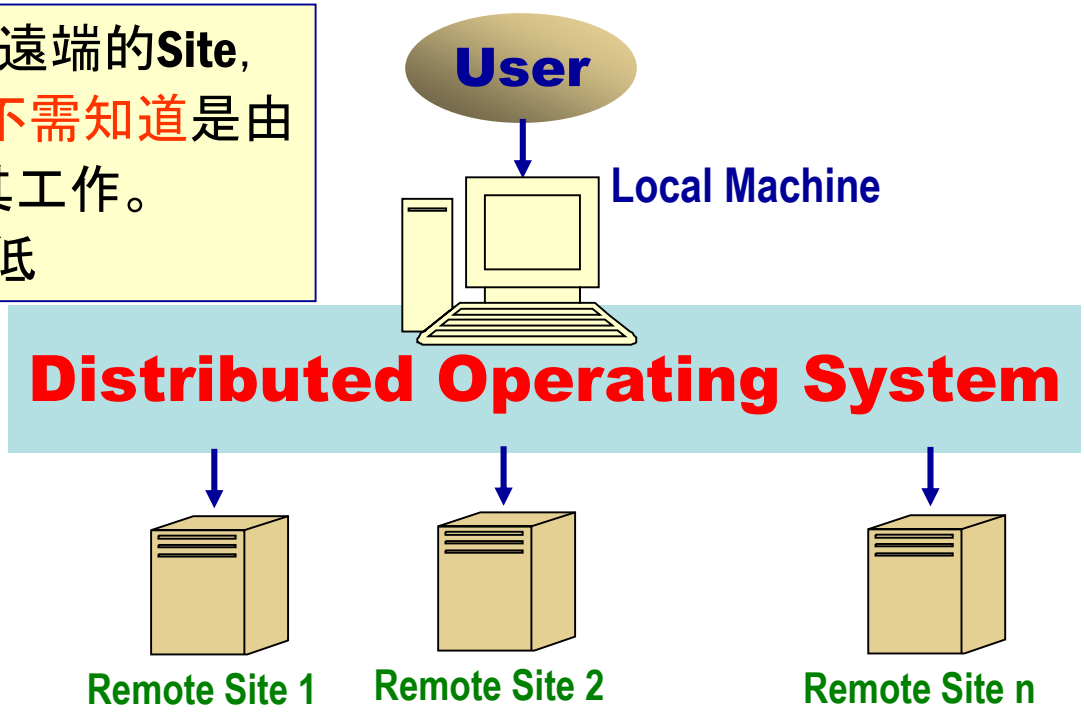
- 提供Network基礎功能，即：O.S. + Network Operation (e.g., Telnet, E-mail, FTP...)。
- User自行決定工作要在哪個機器上執行!!User自主性高。
  - 如果不知道對方主機的Information，則無法使用分散式的功能。



分散式作業系統 (Distributed Operating System)

- 目標是要讓使用者以存取自已資源之方式來存取遠程資源。從一站至另一站之Data與Process的轉移均在分散式作業系統控制之下，提供了只有一個單一作業系統的幻覺。**(最難設計)**
  - 此一特性稱為**Single-System Image (單一系統映像)**或**Virtual Uniprocessor (虛擬單一處理器)**。

由D.O.S來分派遠端的Site，User不知道也不需知道是由哪個Site完成其工作。
⇒ User自主性低



構建分散式系統的理由 (好處)

- **Resource Sharing (資源共享)**
  - 有相同的Resource，不用複製好幾份，∴Cost低。
- **Speed Up (加快計算速度)**
- **Reliability (可靠性)增加**
- **Communication Need (通訊需求)**
  - WWW Service
  - E-mail
  - FTP
  - Telnet

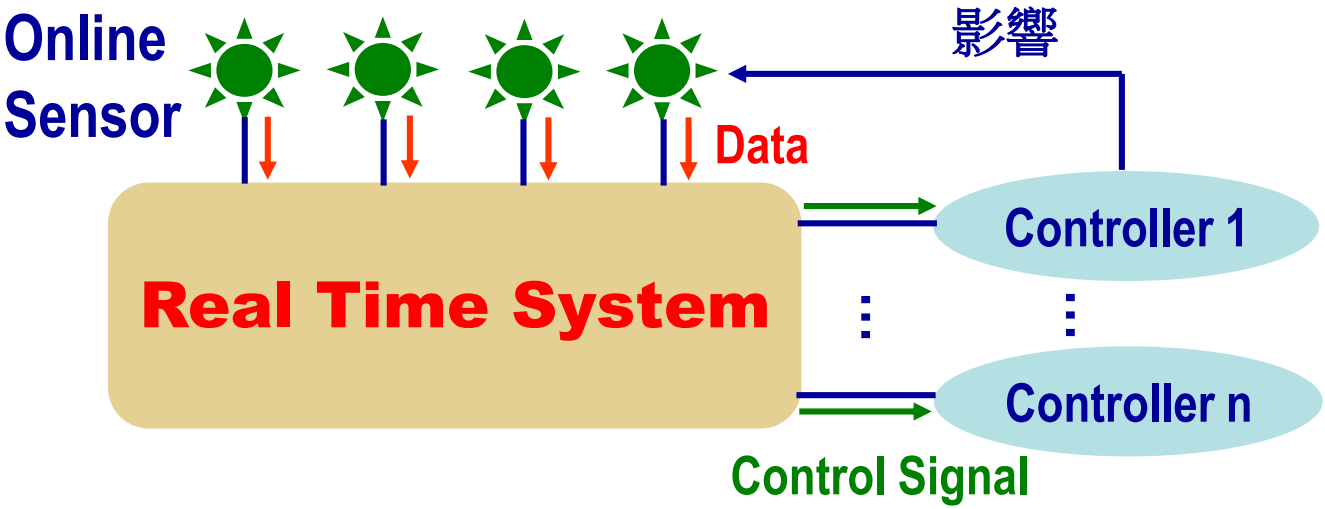
項目	網路作業系統	分散式作業系統	多處理器系統
操作外觀是否像虛擬的單處理機	否	是	是
是否所有電腦都必須執行相同的作業系統	否	是	是
系統中共有幾份作業系統	N	N	1
是否需要共同的網路協定	是	是	否
是否使用單一執行佇列	否	否	是

■即時系統 (Real Time System)

- **Def:** 有著**定義嚴謹的固定時間限制**，電腦在處理工作時，必須在這個定義的時間內完成，否則工作就算失敗 (失效)。
- 分成兩種類型：
  - **硬性即時系統 (Hard Real Time System)**
  - **軟性即時系統 (Soft Real Time System)**

硬性即時系統 (Hard Real Time System)

- **Def:** 對於完成工作的時間有極嚴格的限制。若**Process**未能於規定的時間內完成，則**Process**即屬失效。(意義同即時系統之定義)



- 通常用於:
  - 工廠自動化系統、軍事系統、核能安控...等
  - **O.S.**不用太強，然而在**Application Program**的設計上則非常重要。

- 可以保証重要的工作準時完成。
- 系統內所有的延遲都有所限制。
- 設計硬性即時系統時需注意的事項：
  - Data或Control Signal的傳輸時間、系統處理(運算)時間、硬體設備運作時間...等時間總合需**小於等於Time Constraint**，連帶影響**硬體(Control Device)的選擇**。
  - 大部份的Data及Program皆存在**ROM或RAM**中，輔助儲存體 (e.g., Disk, Tape) 甚少採用或不用。**Virtual Memory**也不使用 (∵**Page Fault的處理時間過長**)。
  - **減少O.S.的干預**，盡量降低Dispatch Latency (分派延遲)。O.S.大多提供基本功能即可 (e.g., Microkernel O.S.)，有的連O.S.都沒有，直接由User Program來處理。
    - **Dispatch Latency (分派潛伏期、分派延遲)**：當O.S.停止一個Process，並開始另一個Process時，會有一些細部工作需要處理。而這些工作所耗用的時間總合即稱為Dispatch Latency 。

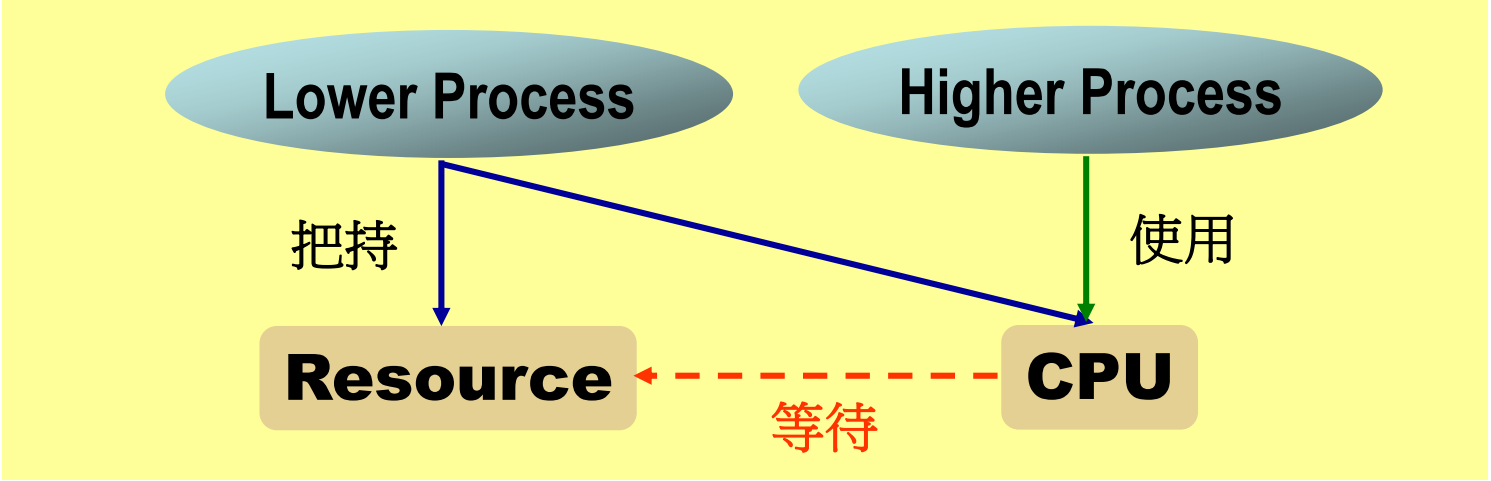
- 一般用途的O.S.皆未提供Hard Real Time特性。
- Hard Real Time System與其它功能系統難以結合。

軟性即時系統 (Soft Real Time System)

- **Def:**
  - 具有即時性的Process應讓它具有最高的優先權 (Priority)，且保証高優先權的Process必須先於所有低優先權的Process完成。
  - 高優先權Process的Priority值會一直維持到其完成為止。
  - e.g., Multimedia System, Virtual Reality... etc.
- 設計軟性即時系統時需注意的事項
  - O.S. (Kernel) 所造成的延遲 (Latency) 時間宜縮短 (即：要有所限制)
  - CPU的排班 (Scheduling)應能支援Priority Scheduling，且不能提供類似 "Aging" 技術
  - 宜避免Priority Inversion (優先權反轉) 問題，可用Priority Inheritance (優先權繼承)來解決。
- Soft Real Time System可與其它系統結合。
- 限制：無法適用於嚴格要求完成時間的系統。

什麼是Priority Inversion問題

- 指一個具有高優先權的Process被一個或多個低優先權的Process給長時間阻擋住，而無法執行其工作。

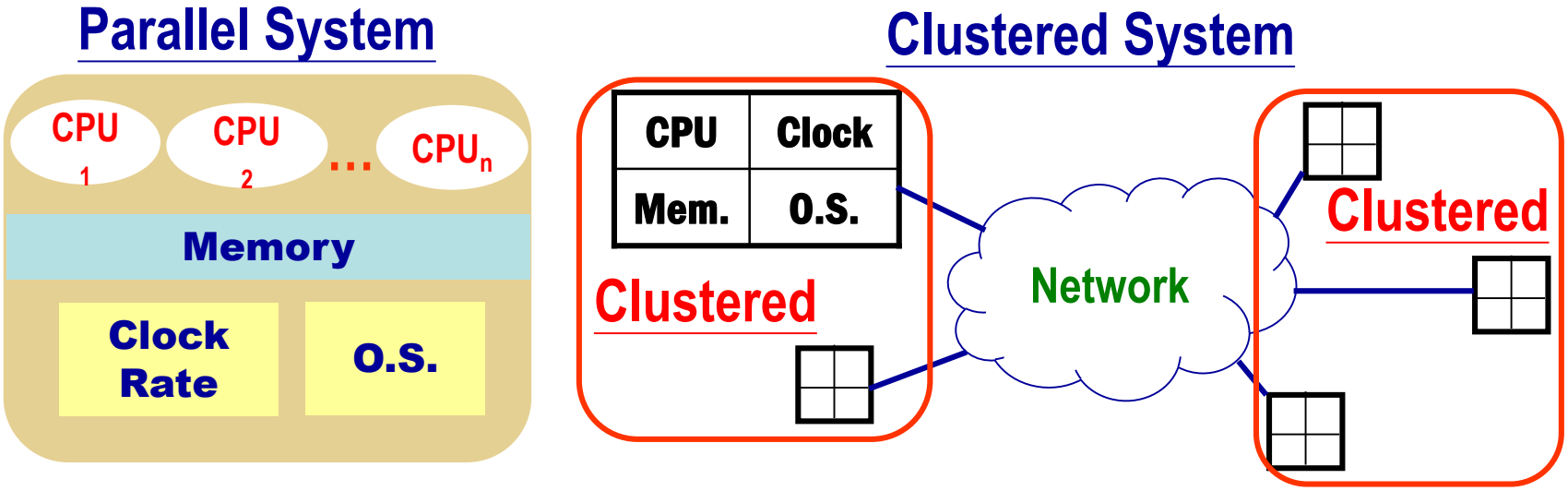


Priority Inversion

- 解決：**Priority Inheritance (優先權繼承)**
  - 暫時性地調高Low Process的Priority到和High Process一樣，等資源釋放後再將它的優先權值調回來。

■集成式系統 (Clustered System)

- Clustered System與Parallel System一樣，集合許多CPU以完成工作。然而Clustered System和Parallel System不同之處在於它們是由兩個或更多個各別系統集結在一起所組成。
- 一般所接受的定義為：Clustered System共享儲存裝置，並且經由LAN連線緊密地連結。



*Clustered:某幾台PC做某一種工作

- Clustered System主要是利用多台獨立的的電腦系統或是工作站來共同完成大型數值的平行計算。
- Clustered System的主要目的在於提供High Availability (高的可利用性)
- 最知名的Clustered System應用例子就是電影鐵達尼號，這部電影用了五百多台的Clustered System來進行電腦動畫的製作。
- 可分為兩個Modes
  - Asymmetric Clustering
  - Symmetric Clustering

非對稱式集成 (Asymmetric Clustering)

- **Def:** 有一台機器處於**Hot Stand-by Mode (熱待機狀態)**，負責監督其它正在執行應用程式的**Server**，若被監督的某台**Working Server**壞了，則此主機就掌管此壞掉的**Server**之**Stage Devices**，並重新執行未完成的應用程式。

對稱式集成 (Symmetric Clustering)

- 多部機器同時執行應用程式，也彼此 (互相)監督對方。
- 具有較高的處理效能 (∵可掌握較多的硬體資源)。

■手持式系統 (Handheld Systems)

- Handheld Systems就是我們現在常見的PDA和手機。
- 在系統設計上，因受限於**實體大小 (Physical Size)**，所以需注意下列的議題 (Issue)：
  - **Limited memory (有限的記憶體):**
    - 目前的手機和PDA沒有虛擬記憶體 (Virtual Memory) 的技術，所以在開發一些軟體時，非常受限於記憶體的實際容量。
    - O.S.和Application需善加利用Memory，一旦程式不再執行，須釋放Memory給系統。
  - **Slow processors (處理速度較慢的處理器):**
    - 因為處理器如果處理速度較快時所需耗費的功率較高，會消耗掉很多的電源，然而手機或是PDA不可能提供這麼大量的電源，所以只能擁有處理速度較慢的處理器。
    - O.S.和Application不要去增加處理器的額外負擔。
  - **Small display screens (較小的顯示器):**
    - 顯示內容需有所限制，所以在網頁開發上會比較受限制。

計算環境 (Computing Environments)

- **Traditional Computing**
  - 先前所介紹的內容，幾乎都是屬於傳統的作業系統的計算環境。更由於網際網路與其他的技術的高度發展，拓展了傳統式計算的範圍。
    - 遠端存取的技術已經成熟，可以架設或是使用各種網路服務。
    - 網際網路式計算、手持式電腦和個人電腦同步以及無線上網的功能，提高系統的**可攜性**。
- **Client-Server System**
  - **Server**可大略地區分成計算伺服器和檔案伺服器兩類
- **Peer-to-Peer Computing**
  - 客戶端與伺服器端彼此無法區分。在系統內所有節點可當作客戶端或伺服器端。

Web-Based Computing

- 基本上，**透過瀏覽器與網際網路伺服器進行**的就是網際網路式計算 **(Web-Based Computing)**。
- 同時，網際網路式計算也帶來了一些新裝置與技術的發展，例如**負載平衡器 (Load Balancers)**，就是將網路連線的負荷分散到網路中相同功能的伺服器上，以提高系統的**效率**。
- 現今的作業系統都已內建網路功能。